NLP Final Project Check-In - Avinash Damania, Abhishek Khare

What we've looked into so far for improving the performance of this QA system is adding Named Entity Recognition (NER) in order to minimize wrong answers and help find answers that overlap more with the question. As noted in the final project document section on linguistic constraints for robustness (Track 1), this especially helps in adversarial datasets like the given SQuAD dataset. Our initial plan was to run NER on a group of the top k potential answers (the different spans from different start/end distributions), and compare this to the result of running NER on the question. For "who" and "where" type questions, we are also planning to prioritize choosing answers if they have a named entity. This could potentially harm performance, however, if the question is something like "where is Joe" and the answer is "in the bathroom", but we intend to at least try this modification and see how it functions on the given datasets.

What we have achieved so far includes the integration of SpaCy into the given QA system, as well as making modifications to the existing utilities and code to enable us to make the modifications we need. One change we made in utils.py was to create a new version of the search_span_endpoints function called topk_span_endpoints. This function returns a list of k optimal answers instead of returning just one. To test this, we changed the call to search_span_endpoints in write_predictions in main.py to instead use topk_span_endpoints[0], which logically should give the best possible answer and lead to the same EM/F1 scores when using the given small baseline model. We found that the scores were the same ({'EM': 47.21, 'F1': 59.81}), so our function was working as expected. We also wrote the code to perform NER on the potential answers and compare them to the question text in write_predictions as well. This code takes a start/end index and finds the sentence(s) surrounding it so we can search for named entities in the area around a potential answer span. We're also going to test contexts of size n around spans because many answers might share the same surrounding sentences. Most of our current code changes are in write_predictions in main.py (see https://github.com/avinashdamania/nlp-final-project)

```
# original way
start_index, end_index = search_span_endpoints(start_probs, end_probs)

# new way using our custom topk_span_endpoints function
max_prob, start_index, end_index = topk_span_endpoints(start_probs, end_probs)[0]
```

The next steps for our project include implementing NER to help in answering the "who" and "where" questions. Another problem we've run into is that we currently can't test the effectiveness of SpaCy because our text is all lowercase, and SpaCy only seems to be able to recognize uppercased named entities. We've written most of the code to perform the named entity recognition, but we need to figure out how to wire together the un-lowercase text with the modifications we've made to the code. Once we've done that, we can run some benchmark tests and see how effective our changes are, and from there we can think about better changes to help with the QA system's performance. One of the changes we had in mind at the moment was weighting answers with common named entities higher, but weighing the probabilities at those words lower, since the answer might be cleaner as one word. For example, for the question "how old is Barack Obama", the answer could be "59" instead of "Barack Obama is 59". This is somewhat specification dependent, as maybe users want the full sentence answer because it's clearer what our system is saying, but we will definitely look into it. We've also already taken into account the fact that the named entities might be outside the answer span in the code we've already written (see the last sentence of the above paragraph). We also need to figure out some sort of "voting" system that takes into account both probabilities and number of common entities between the question and a potential answer instead of entirely choosing answers based on the number of common named entities.